Modelagem em Tempo Real do TRMS Usando Rede Neuro-Fuzzy Evolutiva

Alisson Marques Silva[°⋆], Walmir Matos Caminhas[⋆],
André Paim Lemos[⋆], Fernando Gomide[◇]

[°]Centro Federal de Educação Tecnológica de Minas Gerais, Campus Divinópolis,
[⋆]Programa de Pós-Graduação em Engenharia Elétrica, UFMG,
[◇]Faculdade de Engenharia Elétrica e de Computação - Unicamp,
{alissonmarques,caminhas,andrepl}@cpdee.ufmg.br
gomide@dca.fee.unicamp.br

Resumo Este artigo introduz uma generalização da rede *neuro-fuzzy* evolutiva eNFN (*Generalized* eNFN - G-eNFN) para permitir o cálculo de múltiplas saídas e ilustra sua aplicação na modelagem em tempo real de um sistema de duplo rotor (TRMS - *Twin Rotor MIMO System*) com dois graus de liberdade. O TRMS é um sistema rápido, não linear, instável em malha aberta, variante no tempo e apresenta acoplamento cruzado entre os dois rotores. A modelagem e o controle do TRMS requer alta taxa de amostragem, tipicamente da ordem de milésimos de segundo. Os resultados experimentais mostram que a G-eNFN é rápida, precisa e eficaz na modelagem em tempo real do TRMS. A rede capta rapidamente a dinâmica do sistema e desenvolve modelos precisos de baixo custo computacional. Os resultados sugerem a G-eNFN como uma potencial candidata na modelagem em tempo real de sistemas rápidos, complexos e variantes no tempo.

Palavras-chave: Modelagem em Tempo Real, Sistema *Neuro-Fuzzy* Evolutivo, Sistema de Duplo Rotor.

1 Introdução

A modelagem de sistemas está relacionada principalmente ao desenvolvimento de modelos matemáticos para caracterizar o comportamento de sistemas. Os modelos podem ser desenvolvidos a partir de princípios fundamentais, a partir de dados de entrada/saída, ou ambos. Geralmente, os dados são obtidos experimentalmente, diretamente ou por meio de simulação [1]. Sistemas contemporâneos possuem muitas características que tornam a modelagem matemática tradicional cada vez mais difícil e complexa. Atualmente, os sistemas são caracterizados por alterações frequentes, especialmente nos pontos e condições de operação. Formalismos de modelagem são necessários para facilitar a integração de informações qualitativas e quantitativas, e para combinar com a computação inteligente. A modelagem evolutiva é um exemplo deste tipo de abordagem que tem se mostrado apta para fornecer soluções eficazes de modelagem e controle para sistemas não-lineares variantes no tempo [2].

Hoje, há uma crescente necessidade de monitoramento, detecção e diagnóstico de falhas em sistemas dinâmicos. Basicamente, isto é devido às exigências de alto desempenho de processos industriais reais [3]. A detecção precoce de falhas aumenta a confiabilidade, evita perturbações e/ou interrupções da produção, acidentes e prejuízos financeiros. Uma característica importante e altamente desejável dos métodos de monitoramento, detecção e diagnóstico de falhas é a adaptabilidade [4]. Métodos e modelos podem tornar-se inaplicáveis quando a sua estrutura e parâmetros não se adaptam as mudanças nos processos.

O TRMS (*Twin Rotor MIMO System*) é um *set-up* experimental desenvolvido pela *Feedback Instruments Limited* [5] para simular problemas e desafios de modelagem e controle encontrados na prática. O TRMS é variante no tempo e possui dinâmica complexa, não-linearidade, acoplamento cruzado, fricção, saturação e ruído. Devido a estas características, a modelagem do TRMS a partir de um fluxo de dados é uma tarefa difícil. Abordagens com modelagem adaptativa tem mostrado ser particularmente úteis para lidar com o TRMS [6].

A modelagem e identificação do TRMS foi abordada por diversos autores. Por exemplo, [7] propõe a utilização de um método Quasi-LPV para modelar e estimar parâmetros, enquanto [1] usa um modelo de inferência *neuro-fuzzy* adaptativo (ANFIS) cujo antecedente é ajustado por um algoritmo de otimização por enxame de partículas e os parâmetros do consequente são encontrados por mínimos quadrados recursivos. Abordagens evolucionárias de identificação e modelagem, tais como algoritmos genéticos, enxame de partículas e evolução diferencial são abordados em [8]. As redes neurais artificiais foram exploradas em [9]. Todos esses estudos utilizam dados reais adquiridos do TRMS e os esquemas de modelagem e identificação relatados tem se mostrado bem sucedidos. No entanto, apesar do uso de dados reais, os experimentos e simulações são *off-line*. Eles primeiro executam o TRMS para obter e armazenar os dados. Em seguida, os dados armazenados são utilizados para treinar e avaliar os algoritmos. Este artigo sugere e avalia uma abordagem *neuro-fuzzy* evolutiva para a modelagem em tempo real do TRMS.

A modelagem *fuzzy* evolutiva usa fluxos de dados para, gradualmente, desenvolver a estrutura do modelo simultaneamente com o ajuste dos parâmetros. A modelagem evolutiva fornece um alto nível de adaptação com aprendizado incremental e contínuo [10]. O aprendizado incremental permite um processamento rápido e possui baixo consumo de memória, visto que as amostras de dados são processadas apenas uma vez pelo algoritmo de modelagem e podem ser descartadas posteriormente [11]. Enquanto a aprendizagem facilita a formação contínua de novos conhecimentos, modificando a estrutura e os parâmetros do modelo, ela deve ser capaz de manter e conservar o conhecimento relevante aprendido anteriormente [12]. Os sistemas *fuzzy* evolutivos podem iniciar o processo de aprendizagem a partir do zero e novos conhecimentos podem ser adicionados e/ou excluídos conforme os dados são recebidos.

Os algoritmos evolutivos constituem uma promissora abordagem para o desenvolvimento de modelos adaptativos não-lineares baseados em dados. Estes tem sido amplamente utilizados para construir modelos de sistemas complexos a

partir de amostras de dados. De acordo com [2], há uma crescente demanda para a implantação de sistemas evolutivos. Eles foram bem sucedidos aplicados a problemas de controle [13], classificação [14], identificação, detecção e diagnóstico de falhas [15], previsão e predição [16], para mencionar somente alguns.

Este artigo concentra-se no uso de uma abordagem de rede *neuro-fuzzy* evolutiva para modelar em tempo real um sistema de duplo rotor (TRMS) com múltiplas entradas e múltiplas saídas, com taxa de amostragem na ordem de milésimos de segundo. Mais especificamente, a rede *neuro-fuzzy* evolutiva visa estimar um passo à frente a saída dos controladores PID, ou seja, as voltagens fornecidas para os dois rotores do TRMS. O objetivo é prever em tempo real as saídas dos controladores PID usando como entradas a saída desejada e o ângulo medido a fim de analisar a viabilidade do uso da rede *neuro-fuzzy* como um controlador em tempo real para sistemas não-lineares com dinâmica rápida. Os trabalhos futuros deverão avaliar o seu uso no controle preditivo, controle inverso, monitoramento do estado, e detecção e diagnóstico de falhas em tempo real.

Originalmente introduzida em [17], a eNFN é uma rede *neuro-fuzzy* evolutiva rápida e precisa para modelagem e controle de sistemas. eNFN é um sistema com múltiplas entradas e uma única saída. Neste trabalho sugere-se um algoritmo generalizado para o eNFN (G-eNFN) com múltiplas entradas e múltiplas saídas e sua aplicação na modelagem em tempo real do TRMS. Nos experimentos a G-eNFN começa a modelagem sem conhecimento *a priori* sobre o TRMS. Espera-se que a rede consiga capturar rapidamente a dinâmica das saídas dos controladores.

Após esta introdução o trabalho prossegue como se segue. A próxima seção descreve o TRMS. A Seção 3 introduz a generalização da rede eNFN para múltiplas saídas. A Seção 4 centra-se na modelagem em tempo real do TRMS. Por fim, a Seção 5 conclui o artigo resumindo as suas contribuições e apresentando as questões a serem abordadas no futuro.

2 Sistema MIMO de Duplo Rotor

O Sistema MIMO (*Multiple-Input Multiple-Output*) de Duplo Rotor (TRMS), ilustrado na Figura 1, é constituído por uma haste fixada a um conjunto torre/base. Cada extremidade da haste possui um rotor acionado por um motor de corrente contínua com velocidade variável para controlar o movimento dos eixos vertical (arfagem - *pitch*) e horizontal (guinada - *yaw*). O rotor principal opera no ângulo de elevação (eixo de arfagem - *pitch axis*), enquanto o rotor de calda atua no ângulo de rotação (eixo de guinada - *yaw axis*) para movimentar para direita ou esquerda.

O TRMS em muitos aspectos se assemelha a um helicóptero. Por exemplo, como um helicóptero o TRMS possui um forte acoplamento cruzado entre os dois rotores, uma dinâmica complexa e altamente não-linear. O TRMS possui dois dos três movimentos de um helicóptero, o ângulo de arfagem e o de guinada. Além dos dois movimentos citados, o helicóptero também possui o ângulo de rolagem. No módulo do duplo rotor a posição é controlada por meio da va-

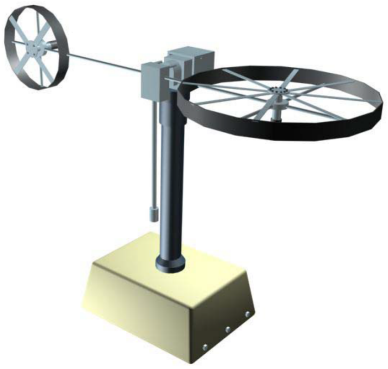

Figura 1. Módulo do Duplo Rotor [5].

riação da velocidade do rotor. No entanto, no helicóptero a força aerodinâmica é controlada pela modificação do ângulo das lâminas do rotor, e a velocidade do rotor é aproximadamente constante.

O Sistema de Duplo Rotor possui duas variáveis de entrada, a voltagem aplicada ao rotor principal u_1 e a voltagem aplicada ao rotor de calda u_2. As variáveis de saída são o ângulo de arfagem (*pitch angle*) Ψ e o ângulo de guinada (*yaw angle*) φ. Outras saídas também podem ser consideradas, como por exemplo, as velocidades angulares, porém elas não são consideradas neste trabalho. A Figura 2 mostra o diagrama simplificado do TRMS. Como pode ser visto, u_1 controla o ângulo de arfagem e u_2 atua no ângulo de guinada. A figura também destaca a dinâmica do acoplamento cruzado entre os movimentos dos dois eixos. Observe que u_1 afeta φ e u_2 perturba Ψ.

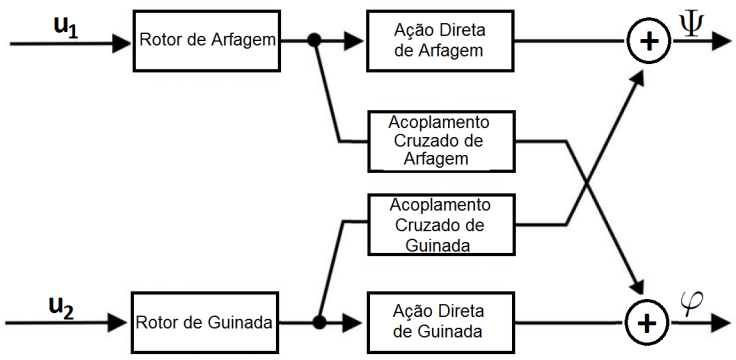

Figura 2. Diagrama simplificado do TRMS.

3 G-eNFN - Rede *Neuro-Fuzzy* Evolutiva com Múltiplas Entradas e Múltiplas Saídas

Esta seção introduz uma generalização para a eNFN [17], chamada de G-eNFN (*Generalized* eNFN). A rede eNFN é uma rede *neuro-fuzzy* evolutiva com n entradas e uma única saída e possui um algoritmo de aprendizagem incremental *single pass*, isto é, o mecanismo de aprendizagem utiliza apenas a amostra de dados atual e todos os cálculos são realizados recursivamente sem a necessidade de armazenar valores passados. O algoritmo de aprendizado incremental, simultaneamente, evolui a estrutura e atualiza os pesos da rede. A estrutura da rede evolui em função do erro de modelagem e dos dados de entrada. Os pesos da rede são atualizados por um algoritmo baseado no gradiente com taxa de aprendizado ideal. A abordagem proposta para a generalização mantém as principais características da eNFN e permite o cálculo de múltiplas saídas.

O algoritmo de aprendizagem da G-eNFN pode ser resumido em 6 passos: 1) inicialização de funções de pertinência, 2) adaptação de contexto, 3) cálculo das saídas, 4) atualização dos pesos, 5) criação de funções de pertinência e 6) eliminação de funções de pertinência. A descrição detalhada destes passos é como se segue.

3.1 Funções de Pertinência

A rede usa funções de pertinência triangulares e complementares. Uma função de pertinência triangular pode ser definida pelo limite inferior, valor modal e limite superior. Denota-se o valor modal da k-ésima função como (b_k). O limite inferior é o valor modal da função adjacente inferior (b_{k-1}), e o limite superior o valor modal da função adjacente superior (b_{k+1}).

Cada uma das n variáveis de entrada é iniciada com o mínimo de duas funções de pertinência $(m_i = 2,\ i = 1, ..., n)$ e dependendo do fluxo de dados e do erro de modelagem, novas funções podem ser criadas e inseridas. O valor modal inicial para as funções de pertinência é dado por $b_{i1} = min_{x_i}$ e $b_{i2} = max_{x_i}$, onde i indexa a variável de entrada, min_{x_i} e max_{x_i} são respectivamente o limite inferior e limite superior do domínio da i-ésima variável, $i = 1, ..., n$.

3.2 Adaptação de Contexto

Na modelagem de sistemas em tempo real podem ocorrer mudanças no ambiente que permitam o surgimento de dados cujos valores estão fora do domínio definido pelos valores máximo (max_{x_i}) e mínimo (min_{x_i}). Portanto, é importante acompanhar e atualizar esses limites. O procedimento para atualizar os valores de max_{x_i} e min_{x_i} é como se segue: *se* $x_{t_i} < min_{x_i}$ *então* $min_{x_i} = x_{t_i}$ *e* $b_{i1} = min_{x_i}$ ou *se* $x_{t_i} > max_{x_i}$ *então* $max_{x_i} = x_{t_i}$ *e* $b_{im_i} = max_{x_i}$.

3.3 Saídas da Rede

Basicamente, a estrutura da rede G-eNFN corresponde a n estruturas tipo Takagi-Sugeno [18] de ordem zero, uma para cada entrada. Os modelos são desacoplados e cada uma das l saídas da rede y_t^l é obtida pela soma das saídas individuais y_{ti}^l. O domínio de cada variável de entrada x_{ti} é granulado em m_i funções de pertinências triangulares e complementares. No entanto, no máximo duas das m_i funções de pertinência são ativadas para uma dada entrada x_{ti} e as saídas individuais são computadas somente para as funções ativas. Cada uma das l saídas da rede é computada como se segue

$$y_t^l = \sum_{i=1}^{n} y_{ti}^l = \sum_{i=1}^{n} (\mu_{A_{ik_i}}(x_{ti}) q_{ik_i}^l + \mu_{A_{ik_i+1}}(x_{ti}) q_{ik_i+1}^l), \tag{1}$$

onde l é o índice de cada saída, t o passo atual, n número de variáveis de entrada, k_i and $k_i + 1$ os índices das duas funções de pertinência ativadas para i-ésima variável de entrada.

3.4 Atualização dos Pesos

Os pesos da rede são atualizados por um algoritmo de aprendizagem incremental e supervisionado usando um procedimento baseado no gradiente com taxa de aprendizado ótima [19] que proporciona um erro de aproximação nulo em um passo. Note que, para a i-ésima entrada x_{ti} em t, somente as funções $\mu_{ik_i}(x_{ti})$ e $\mu_{ik_i+1}(x_{ti})$ são ativas e somente os pesos das funções de pertinência ativas são ajustados. Assim, $q_{ik_i}^l$ e $q_{ik_i+1}^l$ são atualizados por $q_{ik_i}^l = q_{ik_i}^l - \alpha_t(y_t^l - \hat{y}_t^l)\mu_{A_{ik_i}}(x_{ti})$ e $q_{ik_i+1}^l = q_{ik_i+1}^l - \alpha_t(y_t^l - \hat{y}_t^l)\mu_{Aik_i}(x_{ti}+1)$, onde y_t^l é a saída desejada, $\hat{y}_t^l$ é a saída da rede, e a fórmula da taxa de aprendizado $\alpha_t = 1/(\sum_{i=1}^{n} \mu_{Aik_i}(x_{ti})^2 + \mu_{Aik_i+1}(x_{ti})^2)$.

3.5 Criação de Funções de Pertinência

A criação e inserção de funções de pertinência visa refinar a granulação do espaço de entrada para reduzir o erro uniformemente. Para a entrada x_t, calcule, recursivamente, a soma da média do erro $\mu\hat{\mu}_{g_t}$ e a soma da variância do erro $\sigma\hat{\sigma}_{g_t}^2$ de modelagem global de todas as saídas, e estas são divididas pelo número de saídas, como se segue

$$\mu\hat{\mu}_{g_t} = \frac{\sum_{l=1}^{s} \hat{\mu}_{g_t^l}}{s} = \frac{\sum_{l=1}^{s} \hat{\mu}_{g_{t-1}^l} - \beta(\hat{\mu}_{g_{t-1}^l} - e_t)}{s}, \tag{2}$$

$$\sigma\hat{\sigma}_{g_t}^2 = \frac{\sum_{l=1}^{s} \hat{\sigma}_{g_t^l}^2}{s} = \frac{\sum_{l=1}^{s} (1-\beta)(\hat{\sigma}_{g_{t-1}^l}^2 + \beta(\hat{\mu}_{g_t^l} - e_t)^2)}{s}, \tag{3}$$

onde $\hat{\mu}_{g_t^l}$ é a média do erro de cada uma das l saídas, $\hat{\sigma}_{g_t^l}^2$ é a variância do erro de cada uma das l saídas, β é a taxa de aprendizado, s o número de saídas e $e_t = y_{ti}^l - \hat{y}_{ti}^l$.

Encontre a função de pertinência com maior grau de ativação (b_i^*) e calcule recursivamente para a entrada x_{ti} a média do erro local $\hat{\mu}_{b_{ti}} = \hat{\mu}_{b_{t-1i}} - \beta(\hat{\mu}_{b_{t-1i}} - e_t)$.

O limitador τ controla a menor distância permitida entre o valor modal da função criada e o valor modal das funções adjacentes. O limitador τ evita uma granulação muito fina de uma região do domínio de entrada e é calculado por $\tau = \frac{max_{x_i} - min_{x_i}}{\eta}$, onde η é um parâmetro que auxilia no cálculo da menor distância permitida. A distância entre o valor modal das funções de pertinência após uma nova função ser adicionada é obtido como demostrando em (4).

$$\begin{aligned} &Se\ (b_i^* \neq 1\ \ e\ \ b_i^* < m_i)\ então\ dist = \tfrac{(b_{ib_i^*+1} - b_{ib_i^*-1})}{3}, \\ &Se\ (b_i^* = 1) \qquad\qquad\ então\ dist = \tfrac{(b_{ib_i^*+1} - b_{ib_i^*})}{2}, \\ &Se\ (b_i^* = m_i) \qquad\qquad então\ dist = \tfrac{(b_{ib_i^*} - b_{ib_i^*-1})}{2}. \end{aligned} \tag{4}$$

Se ($\hat{\mu}_{b_{ti}^*} > \mu\hat{\mu}_{g_t} + \sigma\hat{\sigma}_{g_t}^2$) e ($dist > \tau$), então uma nova função de pertinência é criada. A criação e inserção de funções de pertinência atualiza a granularização da i-ésima variável do domínio de entrada como se segue:

- **Se** $b_i^* \neq 1$ **e** $b_i^* < m_i$, então a função mais ativa é substituída por duas funções de pertinência cujo valores modais são obtidos por $b_\star^1 = b_{ib_i^*-1} + dist$ e $b_\star^2 = b_{ib_i^*-1} + 2 * dist$,
- **Se** $b_i^* = 1$, então a nova função de pertinência é criada entre a primeira e a segunda e o valor modal da função recém criada é $b_\star^1 = b_{ib_i^*} + dist$,
- **Se** $b_i^* = m_i$, então a nova função é criada e inserida entre a última e a penúltima com valor modal obtido por $b_\star^1 = b_{ib_i^*} - dist$.

3.6 Eliminação de Funções de Pertinência

O procedimento para eliminar regras *fuzzy* (funções de pertinência) usa o conceito de idade da regra [20] que é definida pelo tempo de inatividade de uma função de pertinência. Para cada variável de entrada i, encontre b_i^-, índice da função de pertinência com maior tempo de inatividade. A função de pertinência indexada por b_i^- será eliminada se $idade_{b_i^-} > \omega\ e\ m_i > 2$, onde $idade_{b_i^-}$ computa a instância de tempo que a função indexada por b_i^- esta inativa, e ω é um parâmetro que indica o limiar de tempo de inatividade das funções de pertinência.

Após a eliminação de uma função de pertinência, a granularização do domínio da variável de entrada é realizada como se segue:

- **Se** $b_i^- \neq 1$ **e** $b_i^- < m_i$, então a função indexada por b_i^- é eliminada e os limites das funções adjacentes ajustados,
- **Se** $b_i^- = 1$, então a função indexada por b_i^- é eliminada e o valor modal da função adjacente definido para x_{min_i},

Se $b_i^- = m_i$, então a função indexada por b_i^- é eliminada e o valor modal da função adjacente definido para x_{max_i}.

Uma discussão detalhada sobre o ajuste dos parâmetros β, m, η, ω e seus valores típicos podem ser encontrados em [17].

4 Experimentos

Esta seção apresenta os experimentos computacionais realizados em tempo real utilizando a rede G-eNFN na modelagem do TRMS. No TRMS o controle dos ângulos de arfagem e guinada é realizado por dois controladores PID, um para o movimento de arfagem e o outro para o de guinada. A entrada dos controladores é a diferença entre a posição medida e a posição desejada, e a saída é a voltagem aplicada aos rotores de arfagem e guinada. No experimento será utilizada uma rede G-eNFN com duas saídas, uma para estimar um passo à frente a voltagem aplicada ao rotor de arfagem e a outra para a voltagem aplicada ao rotor de guinada. O objetivo do experimento é analisar a viabilidade e avaliar o desempenho da rede G-eNFN na modelagem em tempo real de sistemas com dinâmica rápida.

No sistema do TRMS o *software* de aquisição de dados e controle é executado em um microcomputador com sistema operacional Windows e utiliza uma placa de aquisição de dados Advantech. As unidades de controle são implementadas em ambientes Matlab e Simulink. O sistema operacional Windows não foi desenvolvido para aplicações em tempo real. Por isso utiliza-se a *Real-Time Workshop* do Matlab [21] para simular o processamento em tempo real.

Na estimação em tempo real um passo à frente, a rede G-eNFN é executada em duas etapas. A primeira etapa visa o ajuste dos parâmetros e a evolução da estrutura. Nesta etapa, a G-eNFN tem como entrada a posição desejada em (t) e as medidas da posição em $(t-1)$. O valor estimado da posição é comparado com a posição medida atual (y_t) e a diferença (erro) é utilizada para atualizar os parâmetros e adaptar a estrutura da rede. Conclui-se esta etapa fixando parâmetros e estrutura da rede. Na segunda etapa estima-se a posição um passo à frente. Nesta etapa, o modelo tem como entrada a posição desejada em $(t+1)$ e as medidas da posição em (t). Utilizam-se estas entradas e os parâmetros e estrutura fixados na primeira etapa para estimar a posição um passo à frente, isto é, $(\hat{y}_{t+1})$. A Figura 3 mostra a estrutura da rede G-eNFN para a estimação em tempo real da voltagem aplicada aos rotores do TRMS.

O tempo de simulação do experimento foi de 100 segundos com a taxa de amostragem de 10^{-3} segundos (1 milésimo de segundo), resultando num total de 100.000 amostras. O ângulo desejado de arfagem é obtido pela soma de três funções seno com amplitude $0,8$, $0,3$, $0,3$ e frequência $0,1$, $0,05$, $0,01$ Hz. Acrescenta-se a essa soma a constante $0,4$. O ângulo desejado de guinada é dado pela soma três funções seno com amplitude $0,1$ e frequência $0,1$, $0,05$, $0,02$ Hz. O desempenho da eNFN na estimação em tempo real será avaliado pelo erro quadrático $E_t = 1/2(y_t - \hat{y_t})^2$.

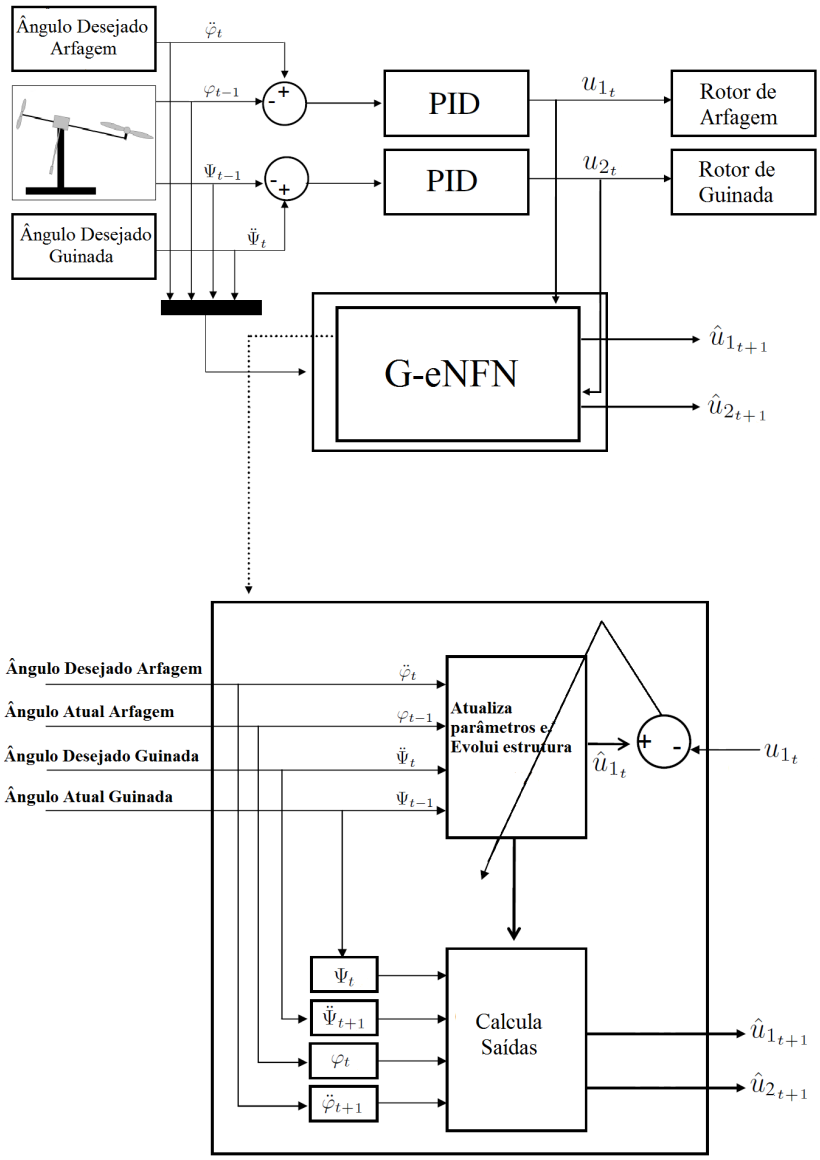

Figura 3. Visão geral da G-eNFN em tempo real.

Neste experimento a rede G-eNFN é empregada para estimar um passo à frente a voltagem aplicada aos rotores de arfagem e guinada. A Figura 4 mostra: (A) a ação de controle real e a estimada para o rotor de arfagem; (B) a ação de controle real e a estimada para o rotor de guinada; (C) a evolução da estrutura da G-eNFN; (D) e (E) o erro quadrático instantâneo E_t para a estimação da ação de controle do rotor de arfagem e guinada, respectivamente. A curva de erro apresentada nas duas figuras confirma a boa precisão da rede G-eNFN na estimação da voltagem aplicada aos rotores do TRMS. Estas figuras também mostram a evolução da estrutura da rede G-eNFN. Note que a rede G-eNFN adapta/evolue sua estrutura de acordo com os dados de entrada e com o erro de modelagem, para manter uma estrutura compacta e uma baixa taxa de erros.

Os experimentos realizados em tempo real no TRMS mostram que a rede G-eNFN é eficiente, isto é, tem boa precisão e é rápida. Além da acurácia e da velocidade, a G-eNFN possui baixo custo computacional. Neste experimento a rede estimou a saída obtida de dois controladores independentes, o que torna o processo de estimação mais complexo.

5 Conclusão

Este artigo introduziu uma generalização para a rede *neuro-fuzzy* evolutiva eNFN (G-eNFN), a qual possibilita o cálculo de múltiplas saídas. A abordagem proposta foi empregada na modelagem em tempo real de um sistema de duplo rotor (TRMS). As voltagens aplicadas aos dois rotores foram estimadas um passo à frente.

Os resultados experimentais mostram que a G-eNFN é eficiente e viável para aplicações em tempo real. Devido à sua baixa complexidade, ao procedimento de aprendizagem e a rápida adaptação, a rede é capaz de modelar um sistema rápido e variante no tempo com eficiência e precisão. Os resultados também destacam a natureza evolutiva da rede. Ela é capaz de desenvolver um modelo adequado para as condições de funcionamento espelhado nos dados de entrada.

Os trabalhos futuros devem ampliar a estrutura da G-eNFN para lidar com a implementação em tempo real dos controladores inversos e preditivos para sistemas complexos variantes no tempo.

Agradecimentos

Os primeiros autores deste trabalho agradecem o apoio da CAPES, CNPq e FAPEMIG. O último autor é grato ao CNPq, processos 304596/2009-4 e 471138/2010-0.

Referências

1. S. Toha and M. Tokhi, "ANFIS modelling of a twin rotor system using particle swarm optimization and RLS," in *Proc. Int. Conf. Cybernetic Intelligent Systems*, 2010, pp. 1–6.

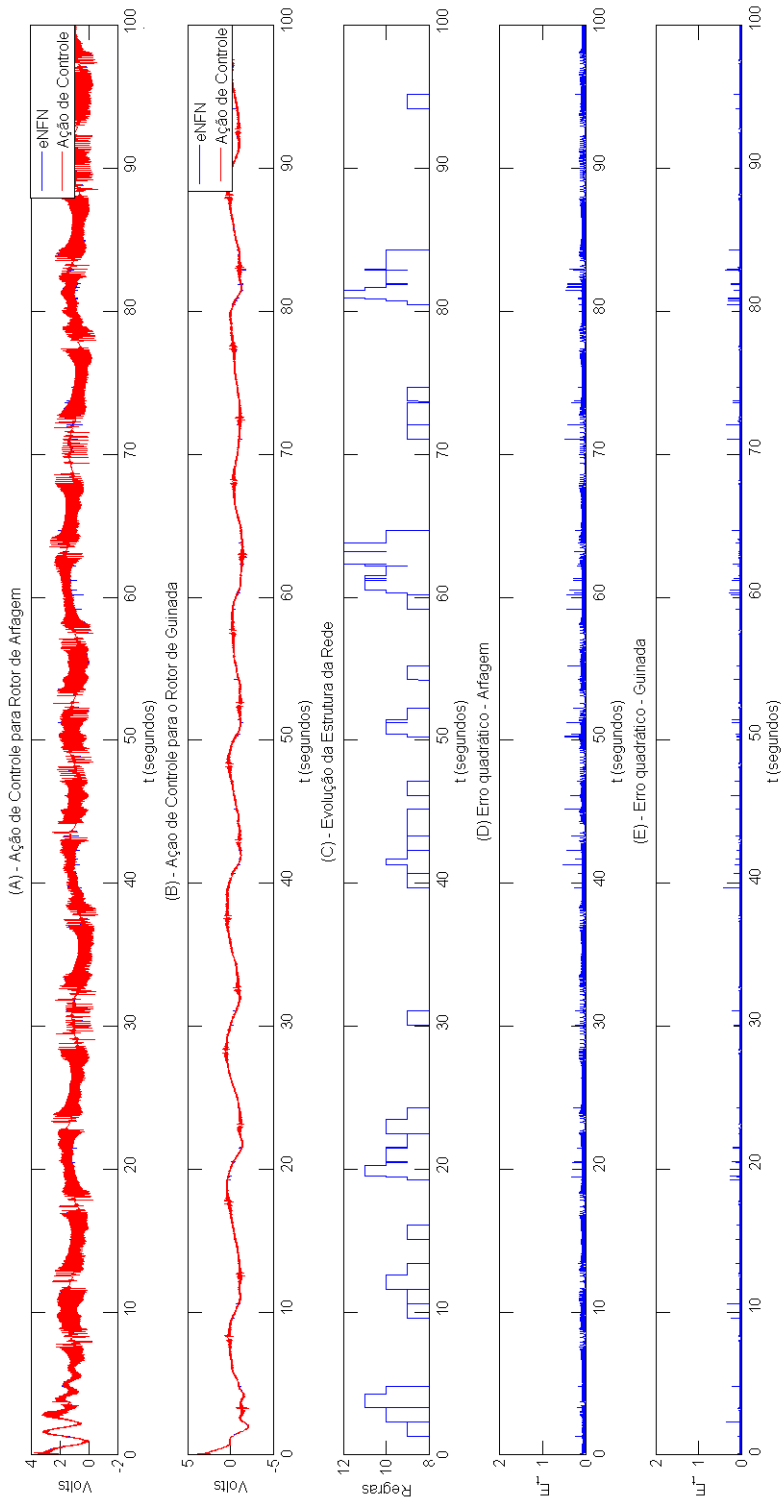

Figura 4. (A) Ação de controle do rotor de arfagem, (B) Ação de controle do rotor de guinada; (C) Evolução da estrutura da eNFN; (D) Erro quadrático na estimação da ação de controle de arfagem; (E) Erro quadrático na estimação da ação de controle de guinada.

2. P. Angelov, D. Filev, and N. Kasabov, "Guest editorial evolving fuzzy systems: Preface to the special section," *Trans. Fuzzy Systems*, vol. 16, no. 6, pp. 1390 – 1392, 2008.
3. A. Lemos, W. Caminhas, and F. Gomide, "Adaptive fault detection and diagnosis using an evolving fuzzy classifier," *Inf. Sciences*, vol. 220, pp. 64–85, 2013.
4. V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri, "A review of process fault detection and diagnosis: Part I: Quantitative model-based methods," *Computers & Chemical Engineering*, vol. 27, no. 3, pp. 293 – 311, 2003.
5. I. Feedback, "Twin rotor MIMO system control experiments, 33-942s," *UK*, 2006.
6. S. Toha and M. Tokhi, "Dynamic nonlinear inverse-model based control of a twin rotor system using adaptive neuro-fuzzy inference system," in *Proc. UKSim European Symposium on Computer Modeling and Simulation*, 2009, pp. 107–111.
7. F. Nejjari, D. Rotondo, V. Puig, and M. Innocenti, "Quasi-LPV modelling and non-linear identification of a twin rotor system," in *Proc. Mediterranean Conf. Control Automation*, 2012, pp. 229–234.
8. B. Subudhi and D. Jena, "Nonlinear system identification of a twin rotor MIMO system," in *Proc. Region 10 Conference*, 2009, pp. 1–6.
9. F. Aldebrez, I. Darus, and M. Tokhi, "Dynamic modelling of a twin rotor system in hovering position," in *Proc. Int. Symp. Control, Communications and Signal Processing*, 2004, pp. 823–826.
10. L. Maciel, A. Lemos, F. Gomide, and R. Ballini, "Evolving fuzzy systems for pricing fixed income options," *Evolving Systems*, vol. 3, no. 1, pp. 5–18, 2012.
11. E. Lughofer, "On-line assurance of interpretability criteria in evolving fuzzy systems - achievements, new concepts and open issues," *Inf. Sciences*, vol. 251, no. 0, pp. 22–46, 2013.
12. S. Tung, C. Quek, and C. Guan, "eT2FIS: An evolving type-2 neural fuzzy inference system," *Inf. Sciences*, vol. 220, no. 0, pp. 124 – 148, 2013.
13. F. Smith and A. Tighe, "Adapting in an uncertain world," in *Proc. Int. Conf. on Systems, Man and Cybernetics*, vol. 6, 2005, pp. 5958 – 5963.
14. P. Angelov, X. Zhou, D. Filev, and E. Lughofer, "Architectures for evolving fuzzy rule-based classifiers," in *Proc. Int. Conf. Systems, Man and Cybernetics*, 2007, pp. 2050–2055.
15. J. Iglesias, P. Angelov, A. Ledezma, and A. Sanchis, "Modelling evolving user behaviours," in *Proc. Workshop on Evolving and Self-Developing Intelligent Systems*, 2009, pp. 16 – 23.
16. E. Lughofer, V. Macian, C. Guardiola, and E. Klement, "Identifying static and dynamic prediction models for NOx emissions with evolving fuzzy systems," *Applied Soft Computing*, vol. 11, no. 2, pp. 2487 – 2500, 2011.
17. A. Silva, W. Caminhas, A. Lemos, and F. Gomide, "A fast learning algorithm for evolving neo-fuzzy neuron," *Applied Soft Computing*, vol. 14, Part B, no. 0, pp. 194 – 209, 2014.
18. T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *Trans. Systems, Man and Cybernetics*, vol. 15, no. 1, pp. 116 – 132, feb. 1985.
19. W. Caminhas and F. Gomide, "A fast learning algorithm for neofuzzy networks," in *Proc. Inf. Processing and Management of Uncertainty in Knowledge Based Systems*, vol. 1, no. 1, 2000, pp. 1784 – 1790.
20. P. Angelov and D. Filev, "Simpl_eTS: A simplified method for learning evolving Takagi-Sugeno fuzzy models," in *Proc. Int. Conf. Fuzzy Systems.*, may 2005, pp. 1068 – 1073.
21. I. MathWorks, "Real-time workshop 7 users guide," *Natick, MA, USA*, 2009.